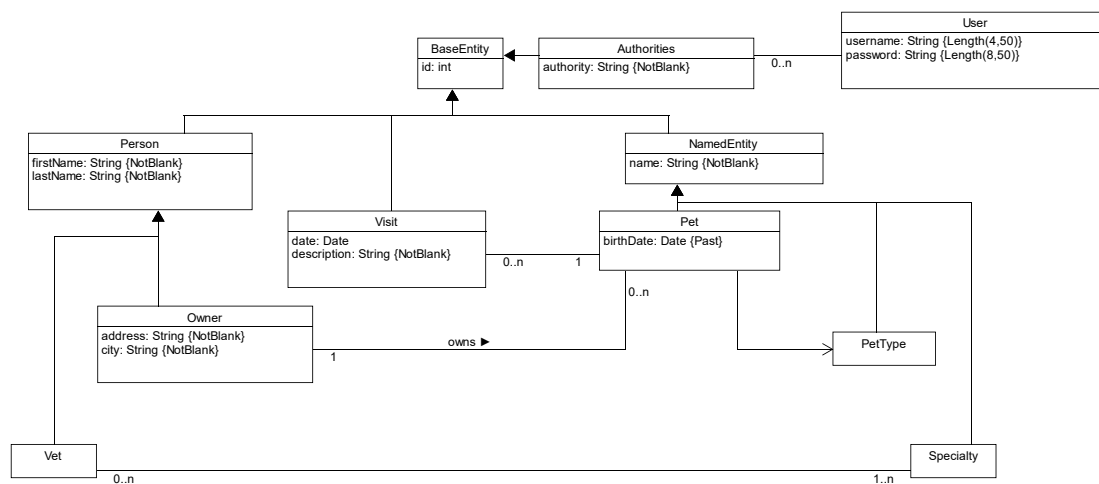


## Boletín de ejercicios intermedios para DP1 2020-2021

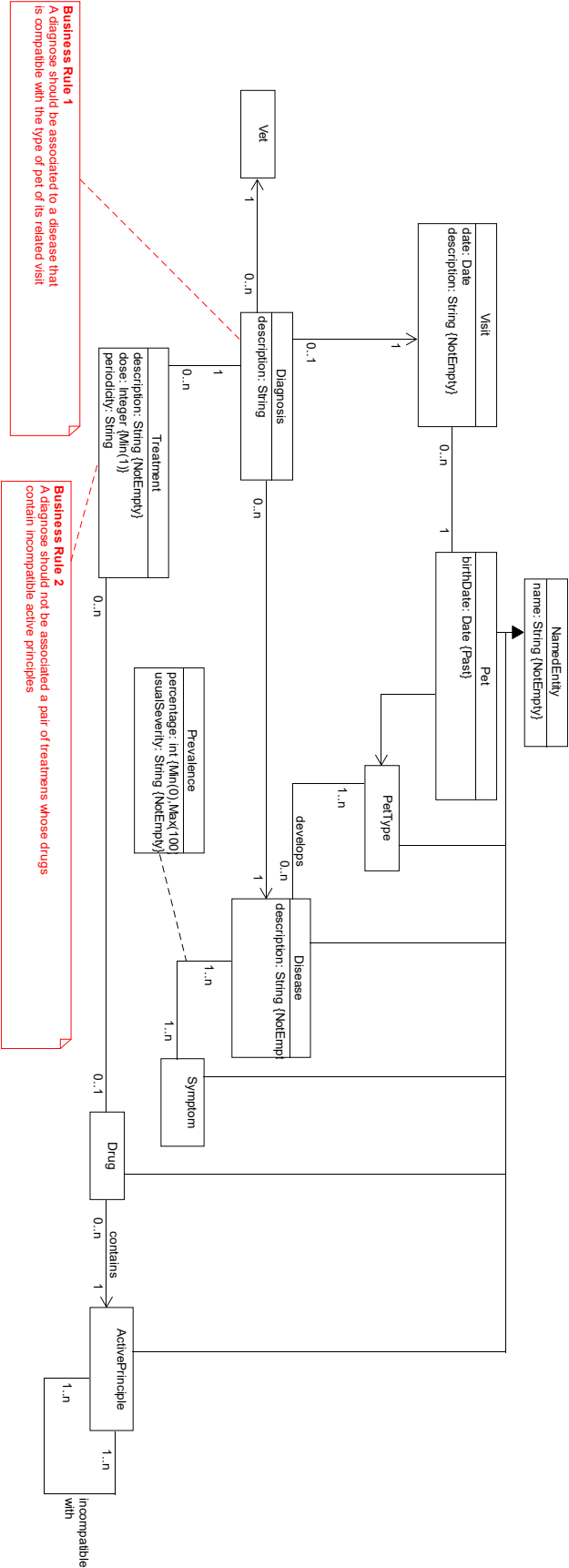
### Introducción y objetivos

Este es el segundo boletín de ejercicios de familiarización con la tecnología de DP1. Si no has realizado previamente el boletín previo de ejercicios básicos, te recomendamos que realices dichos ejercicios antes que los del presente boletín. Los ejercicios de este boletín están asociados al repositorio del [proyecto plantilla de la asignatura](#), y las soluciones están disponibles a través de diversas commits y en su versión final en la rama “japarejo”.

Durante los ejercicios del presente boletín implementaremos un sistema de vademécum y gestión de las consultas de los veterinarios en la clínica de mascotas. Para ello, pasaremos de tener el siguiente diagrama de dominio inicial:



A implementar el diagrama de dominio final que se muestra a continuación:



En este boletín, continuamos con esta tarea a partir del estado de repositorio que teníamos al final del boletín de ejercicios 1. Si no tienes a tu disposición el proyecto en ese estado, puedes partir de dicho estado siguiendo los pasos del ejercicio 0. Si tienes resuelto localmente el primer boletín en tu repositorio, omite el ejercicio 0 y continúa trabajando en el ejercicio 1. Continuaremos la implementación del vademecum paso a paso a través de diversos ejercicios guiados y con soluciones disponibles para el usuario.

## Ejercicios

### Ejercicio 0. (RESUELTO) Creación del repositorio y posicionamiento en el estado inicial.

- a. Inicializaremos el repositorio en un estado conocido con los siguientes comandos:
  - i. Clonado del repositorio:  
`git clone https://github.com/gii-is-DP1/spring-petclinic`
- b. Crearemos una rama llamada my-intermediate-exercises para la trabajar en los ejercicios a partir del estado correspondiente del repositorio (), lo que nos proporcionará un estado inicial adecuado:
  - i. `git checkout -b my-intermediate-exercises 42ffce5f32336732bcaa2948d7e170ab996b23b9`
  - ii. Debería aparecer el siguiente mensaje:  
`Switched to a new branch 'my-intermediate-exercises'`
- c. Comprobación de que el estado inicial es correcto:
  - i. Arrancar la aplicación:
    1. Usando Maven:  
`mvn install`  
`mvn spring-boot:run`
    2. Desde Eclipse:  
Seleccione la clase PetclinicApplication del paquete `org.springframework.samples.petclinic`, y pulse clickDerecho → Run as → Java Application  
Bien en la consola de eclipse bien en la línea de comandos debe aparecer un mensaje similar a este:  

```
INFO 19040 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
```

```
INFO 19040 --- [ restartedMain] o.s.s.petclinic.PetclinicApplication : Started PetclinicApplication in 15.935 seconds (JVM running for 17.525)
```
  - ii. Navegar a <http://localhost:8080> y comprobar que se muestra la página de inicio de la aplicación. Ejecutar el caso de uso de listado enfermedades y comprobar que se pueden editar eliminar y crear enfermedades.

### Ejercicio 1. Creando los diagnósticos asociados a las visitas.

(Ejercicio sobre relaciones: 0..1:1 y 0..1:N entre entidades)

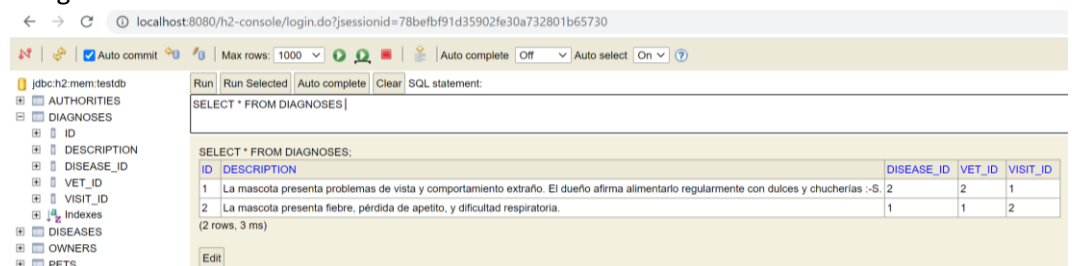
Commit Solución: [d2cba2641ca916a4f87055024b45d274d71f2b9b](#)

Historia de usuario: “Como veterinario, deseo que el sistema me permita reflejar el diagnóstico de la enfermedad que sufren las mascotas (si la hubiera) para las distintas visitas que realicen a la clínica, para mejorar la trazabilidad de las enfermedades de la mascota y ayudar a futuros diagnósticos y visitas”

- Cree una clase java en el paquete `org.springframework.samples.petclinic.model` llamada `Diagnose` que extienda a `BaseEntity`.
- Anote la clase con las anotaciones correspondientes a una entidad.
- Añada un atributo denominado `description` a la entidad, con una longitud máxima de 1024 caracteres.
- Añada una relación 0..1:1 con `Visit`.
- Añada una relación 0..1:N con `Disease`.
- Añada una relación 0..1:N con `Vet`.
- Modifique el script de inicialización de la BD para que incluya los datos de las siguientes visitas:

Id	Visit_Id	Disease_id	Vet_id	Descripción
1	1	2	2	La mascota presenta problemas de vista y comportamiento extraño. El dueño afirma alimentarlo regularmente con dulces y chucherías :-S.
2	2	1	1	La mascota presenta fiebre, pérdida de apetito, y dificultad respiratoria.

- Ejecute la aplicación y acceda a la consola de administración de H2 ( en la url <http://localhost:8080/h2-console/>, recuerde que la url de la BD suele ser `jdbc:h2:mem:testdb` y puede necesitar cambiarla en el formulario de login) para comprobar que la tabla se ha creado y que los datos están efectivamente almacenados en ella. Debe ver una imagen similar a la siguiente:



## Ejercicio 2. Página de Historia clínica de las macotas (Consultas personalizadas)

Commit Solución: [5dc67066e7d10cb46ab575f447a71ba9c75fc4c9](#)

Ojo! Somos humanos 😞 y hay algunos cambios en el commit de la solución posterior que forman parte de esta, por lo si lo necesitáis que os recomendamos ver también: [a1e87375f5842ba32073b93e75808e545fadd868](#)

**Historia de usuario:** “Como veterinario, deseo que el sistema me permita *visualizar el diagnóstico* de las enfermedades que sufren las mascotas (si la hubiera) para las distintas visitas que realicen a la clínica, para mejorar la trazabilidad de las enfermedades de la mascota y ayudar a futuros diagnósticos y visitas”

- a. Cree un repositorio llamado `DiagnosesRepository` en el paquete `org.springframework.samples.petclinic.repository`.
  - i. Esta interfaz debe extender `CrudRepository<Diagnose, Integer>`.
  - ii. Incluya un método para obtener todos los diagnósticos llamado `findAll` (devolvería una `Collection` de `Diagnose`).
  - iii. Incluya otro método, llamado `findByPetId` que devuelva una `Collection` de `Diagnose`. Este método de tomar un parámetro entero llamado `petId` que representa el identificador de la mascota para la que deseamos obtener el listado de diagnósticos de su historia clínica. Este método tendrá asociado una consulta personalizada. Recuerde que debe usar la anotación `@Query` para especificar la consulta personalizada y que el parámetro del método debe estar anotado a su vez como: `@Parameter("<nombre del parámetro en la consulta>")` para poder usar dicho parámetro en la de la consulta. Los resultados deben estar ordenados por fecha de visita (esto último no está incluido en la solución, se deja como ejercicio).
- b. Cree un servicio llamado `DiagnosesService` en el paquete `org.springframework.samples.petclinic.service`. Inyecte el repositorio de diagnósticos en el servicio y cree un método que devuelva todos los diagnósticos y otro que devuelva los diagnósticos de una mascota invocando al repositorio para ello. No olvide anotar la clase como un servicio.
- c. Cree un nuevo controlador llamado `DiagnoseController` en el paquete `org.springframework.samples.petclinic.web`. Cree un método que:
  - i. Responda a peticiones de tipo GET en la url `"pets/{petId}/history"`.
  - ii. Inyecte un objeto de tipo `DiagnosesService` como atributo en el controlador para poder obtener los diagnósticos de la mascota.
  - iii. Busque todos los diagnósticos de la mascota y se los pase a la vista que se describe a continuación.
- d. Cree una vista llamada `ClinicalHistory.jsp` en la carpeta `/src/main/webapp/WEB-INF/jsp/pets`. Esta vista debe mostrar una tabla con los diagnósticos nombre de la enfermedad y la descripción como columnas. Use como plantilla el listado de owners (copiando el contenido y eliminando/modificando los elementos). El

resultado debería asemejarse a:

  HOME  FIND OWNERS  VETERINARIANS  VADEMECUM  ADMIN 

**Max Clinical History**  
Name: Max  
Birth Date: 2012-09-04  
Type: cat  
Owner: Jean Coleman

Date	Vet	Disease	Vet comments
2013-01-02	JamesCarter	COVID-19	La mascota presenta fiebre, pérdida de apetito, y dificultad respiratoria.

- e. Arranque y visualice la historia de Max en <http://localhost:8080/owners/6/pets/8/history>

### Ejercicio 3. Creación y visualización de diagnósticos en la tabla de visitas de los owners

Commit Solución: [a1e87375f5842ba32073b93e75808e545fadd868](https://github.com/1e87375f5842ba32073b93e75808e545fadd868)

**Historia de Usuario:** “Como veterinario, deseo que el sistema me permita crear el diagnóstico de la enfermedad que sufren las mascotas durante las visitas, para mejorar la trazabilidad de las enfermedades de la mascota y ayudar a futuros diagnósticos y visitas”

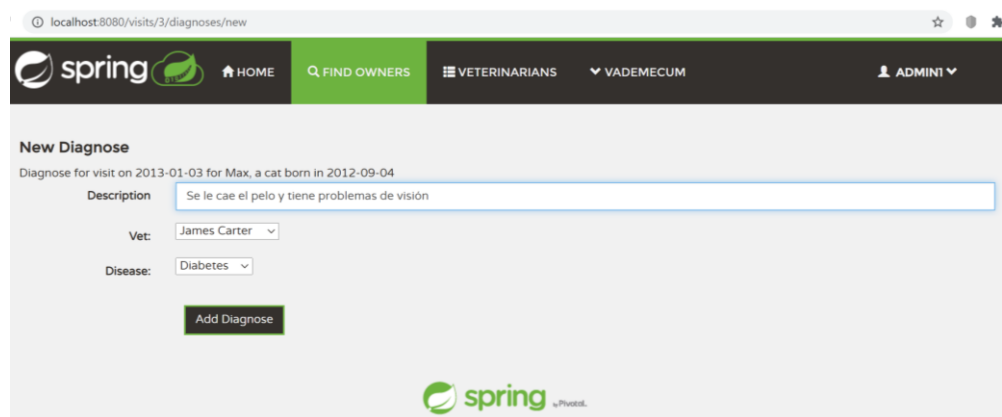
- a. Cree un nuevo método en el servicio de diagnósticos que te devuelva un mapa que tenga como clave las visitas de una lista mascotas que nos pasen por parámetro y como valor el diagnóstico que tenga asociado cada visita. Este método debe invocar al repositorio de diagnósticos para obtener los diagnósticos de cada mascota.
- b. Añada dos métodos al controlador de diagnósticos:
  - i. Un método que responderá a peticiones GET sobre la url /diagnoses/<X>/edit donde <X> es el identificador de diagnóstico a editar. Este método creará un diagnóstico nuevo y lo pasará a la vista “diagnoses/CreateOrUpdateDiagnoseForm”.
  - ii. Un método que recibirá los datos del formulario de creación/edición de diagnósticos. Para ello responderá a peticiones POST en la misma URL. Concretamente el método tomara como parámetros: el id de la visita este será un parámetro de url (@PathVariable), la descripción del diagnóstico, el id de la enfermedad a asociar, y el id del veterinario que emite el diagnóstico, todos esos serán valores obtenidos mediante el formulario. Recuerde que puede obtener los valores del formulario a través de los parámetros del controlador usando la anotación @RequestParamer("<nombre del campo>"). Ese método deberá:
    1. Comprobar que la visita para la que se pretende crear el diagnóstico existe (a partir de su id). Y si no redirigir a la página de inicio con un mensaje de error.
    2. Crear el nuevo diagnóstico y asignarle la descripción.
    3. Obtener y asignar al nuevo diagnóstico la visita, el veterinario y la enfermedad seleccionadas. Para ello deberá hacer uso de los servicios de las entidades correspondientes. Puede obtener los datos de los Identificadores a partir de los campos del formulario mediante las siguientes anotaciones.

```
public String createDiagnose(@PathVariable("visitId")int visitId,  
    @RequestParam("vet") int vetId,@RequestParam("disease") int diseaseId,  
    @RequestParam("description") String description, ModelMap model) {
```

4. Salvar el diagnóstico y redirigir al usuario a la vista de historia de la mascota para que pueda verse el nuevo diagnóstico asociado a la mascota.
- c. Cree la vista de edición de diagnósticos, el fichero debe llamarse CreateOrUpdateDiagnoseForm.jsp y debe estar en la carpeta src/main/webapp/WEB-INF/jsp/diagnoses. Use el contenido del formulario de edición de enfermedades como punto de partida. Recuerde que para la edición de las relaciones 1:N (con Veterinario y Enfermedad, la visita debería ser inmutable a través de este formulario) puede usar un drop-down usando la siguiente estructura de código para generar las opciones de la etiqueta select:  
<form:options itemValue="<nombreDelCampoId>"

```
itemLabel="<nombreDelCampoDeEtiqueta>"
items="${< colección de objetos entre los que elegir>}" />
```

La vista debería tener un aspecto similar al que se muestra en la siguiente imagen:



- d. Inserte un nuevo enlace en la vista de owners (fichero OwnerList.jsp en la carpeta \src\main\webapp\WEB-INF\jsp\owners) que muestre:
  - i. Si el diagnóstico asociado a la visita existe un enlace para editarlo/visualizarlo cuyo (texto sea el nombre de la enfermedad y entre paréntesis el veterinario que lo diagnosticó).
  - ii. Si el diagnóstico asociado a la visita no existe un enlace para su creación. Puede usar las etiquetas para ello a partir de la línea 73 :



```
<c:choose>
  <c:when test="${diagnoses[visit]==null}">
    <spring:url value="/visits/{visitId}/diagnoses/new"
    var="newDiagnoseUrl">
    <spring:param name="visitId" value="${visit.id}"/>
    </spring:url>
    <a href="${fn:escapeXml(newDiagnoseUrl)}">Create
    Diagnose</a>

  </c:when>
  <c:otherwise>
    Diagnose:
    <c:out value="${diagnoses[visit].disease.name}"/>
  </c:otherwise>
</c:choose>
```

- i. Tenga en cuenta que deberá modificar el controlador de owners para hacer uso del servicio de diagnósticos e invocar el método creado en el apartado a) del presente ejercicio para obtener la información de los diagnósticos asociados a cada visita (poder decidir si mostrar o no los enlaces correspondientes en la vista). Para ello deberá inyectar el servicio de diagnósticos (declárelo como opcional con el parámetro required a false en la anotación @Autowired, y tenga en cuenta que en algunos contextos este servicio puede ser nulo al intentar usarlo, como por ejemplo algunas de las pruebas ya creadas, que no proporcionan un doble de pruebas para este servicio, esto se explicará a lo largo del curso más adelante). Además deberá pasar el mapa de visitas a diagnósticos a la vista. El aspecto de la nueva vista de detalles de propietario debería ser similar al que se muestra en la siguiente imagen:



← → ↻ localhost:8080/owners/6/ ☆ ⚙ ⚙ ⚙ ⚙

  [HOME](#) [FIND OWNERS](#) [VETERINARIANS](#) [VADEMECUM](#) [ADMIN](#)

### Owner Information

Name	Jean Coleman
Address	105 N. Lake St.
City	Monona
Telephone	6085552654

[Edit Owner](#) [Add New Pet](#)

### Pets and Visits

Name	Birth Date	Type	Visit Date	Description
Max	2012-09-04	cat	2013-01-03	neutered Diagnose: Diabetes
			2013-01-02	rabies shot Diagnose: COVID-19
			<a href="#">Edit Pet</a>	<a href="#">Add Visit</a>
Samantha	2012-09-04	cat	2013-01-04	spayed <a href="#">Create Diagnose</a>
			2013-01-01	rabies shot Diagnose: Diabetes
			<a href="#">Edit Pet</a>	<a href="#">Add Visit</a>

- ii. La manera de conseguir saber si una visita tiene un diagnóstico o no parece demasiado complicada. ¿Cómo cambiaría usted el diagrama de clases (y consecuentemente la implementación) que describe el modelo de información del sistema para simplificar esto?

#### Ejercicio 4. Automatizando el binding de los datos en la creación de mascotas.

Commit Solución: [69e03552ad32acf39f5c9dc44e5389017f2f51b8](#)

Este es un ejercicio un tanto especial, porque no nos centraremos en dar soporte a una nueva historia de usuario, sino que nos dedicaremos a mejorar la implementación que se hizo en el ejercicio anterior. Concretamente, vamos a evitar tener que procesar manualmente y obtener todas las entidades asociadas a las relaciones 1:N de los diagnósticos. Para ello activaremos el uso de un conversor de tipos condicional genérico que se encuentra implementado en la clase `GenericIdToEntityConverter` del paquete `org.springframework.samples.petclinic.configuration`. Esta clase se encarga de transformar los ids en entidades lanzando consultas a la BD cuando el binding de los datos que vienen en el formulario necesite transformar un entero en un objeto que extiende a `BaseEntity`. Para ello usa un objeto denominado `EntityManager`, que representa a todos los efectos una conexión con la BD y nos permite recuperar una entidad mediante el método `find`, que toma como parámetros el tipo de la entidad y su id.

- a. Para que esta clase haga su trabajo tenemos que registrarla en el subsistema de transformación y formateo de tipos de Spring. Ya le hemos proporcionado la configuración necesaria, usted solamente necesitará activarla. Para ello, añada la anotación `@Configuration` a la clase `WebConfig` que se encuentra en el paquete `org.springframework.samples.petclinic.configuration`
- b. Además, deberá modificar tanto la signature como la implementación del método `CreateDiagnose`. La signature será:

```
public String createDiagnose(  
    @PathVariable("visitId") int visitId,  
    @Valid Diagnose diagnose,  
    BindingResult result, ModelMap model) {
```

- c. En la implementación sencillamente recuperamos automáticamente la entidad `Visit` a partir de su id, se la asociamos al diagnóstico y grabamos el diagnóstico (nótese que los atributos del diagnóstico a crear ya han sido establecidos mediante el binding automático de Spring, incluidas las relaciones con las entidades).
- d. Ejecute en modo depuración la aplicación y ponga un punto de ruptura en el controlador para comprobar que se han asignado los valores apropiados a los atributos del `Diagnose` que nos proporcionan como parámetro en el método.
- e. ¿Puede simplificarse aún más el código del método controlador si modificamos ligeramente la signature del método? Observe la signature del método en el commit de la solución y la implementación proporcionada del método. ¿En qué cambia respecto de las instrucciones proporcionadas?

## Ejercicio 5. Validación de los diagnósticos

Commit Solución: Clase Diagnose en [d2cba2641ca916a4f87055024b45d274d71f2b9b](#)

**Historia de Usuario:** “Como administrador, deseo que el sistema obligue a los veterinarios a reflejar un mínimo de información (*al menos 10 caracteres*) en la descripción del diagnóstico de la enfermedad que sufren las mascotas durante las visitas, para mejorar la trazabilidad de las enfermedades de la mascota y ayudar a futuros diagnósticos y visitas”

- a. Introduzca la restricción respecto del tamaño mínimo de la descripción del diagnóstico usando la anotación `@Size()` y su propiedad `min` en la clase `Diagnose`.
- b. Si no ha usado la etiqueta `<petclinic:inputField>` no olvide incluir la información de validación con las etiquetas `<spring:bind>` o `<form:errors>`.

**Commit Solución:** [c9bfdf9dd991cf4b8f3fa34a5188d35d3769762b](#)

- a. Como parte de este ejercicio vamos a crear una nueva etiqueta personalizada llamada `<petclinic:richTextArea>` que nos permitirá incluir cómodamente en nuestros formularios áreas de texto con editores ricos (que permite editar html con una interfaz similar a los editores de documentos tipo MS Word).
  - i. Cree el fichero `richTextArea.tag` en la carpeta `src/main/webapp/WEB-INF/tags` (puede hacerlo creando una copia de `inputField.tag` y renombrando el fichero para mayor comodidad).
  - ii. Reutilizaremos todo el código de los mensajes de validación por lo que lo único que necesitamos cambiar por ahora es la etiqueta de la línea 16 sustituyendo `form:input` por `form:textarea`.
  - iii. Cambie las etiquetas asociadas a la edición de las descripciones en los formularios de creación/edición que enfermedades y diagnósticos.
  - iv. Compruebe que los formularios ya muestran un área de texto simple para editar las descripciones.
- b. Incluiremos el editor de texto rico como librería externa y cargaremos sus css y javascript en nuestra página.
  - i. Para incluir la dependencia de la librería del editor rico y que Maven se encargue de descargar los ficheros debemos incluir el siguiente contenido dentro de la etiqueta `<dependencies>` del fichero `pom.xml`:

```
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>ckeditor</artifactId>
    <version>4.14.0</version>
</dependency>
```

Le recomendamos integrar esta dependencia en la línea 116 para que estén todas las dependencias de librerías con CSS y Javascript (llamadas webjars) juntas. No olvide hacer un Maven update y un Maven install.
  - ii. Modifique el fichero de la etiqueta personalizada `richTextArea` para que a partir de la línea 25 integre el siguiente código javascript:

```
<script>
    CKEDITOR.replace( '${name}' );
</script>
```

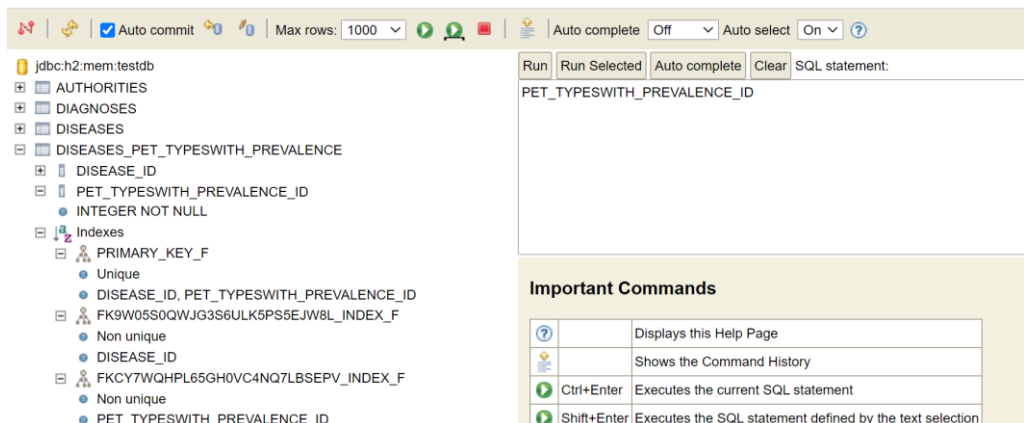
Este código se encarga de generar el editor rico buscando un textarea cuyo name sea el que se le pasa por parámetro a la función `replace`.
  - iii. Añada la librería javascript del editor (que estará disponible en la url) como un script enlazado en la etiqueta `htmlHeader.tag`.
- c. Arranque la aplicación y compruebe que le permite editar las descripciones como texto rico, y que en la historia clínica de la mascota aparece dicho texto adecuadamente.

## Ejercicio 7. Categorizando enfermedades por tipos de mascota (Relaciones n:n).

Commit Solución: [99238ce45ff518611964fec0c637c75aed9efdf](#)

**Historia de Usuario:** “Como veterinario, deseo que el sistema me permita especificar y conocer los tipos de mascotas en los que puede darse cada enfermedad, para evitar diagnósticos erróneos.”

- a. Añada una relación N:N entre Disease y PetType con navegabilidad unidireccional desde Disease.
  - i. Cree una propiedad de tipo Set<PetType> en la clase Disease llamada petTypesWithPrevalence. Indique además que dicho conjunto no puede estar vacío.
  - ii. Anote dicha propiedad como una relación N:N.
- b. Arranque la aplicación y acceda a la consola de h2 (puede ver cómo se hace en el ejercicio 1 del boletín de ejercicios básicos). Compruebe que se ha creado una nueva tabla en la bd con claves ajenas hacia Disease y PetType. La consola debería mostrar un resultado similar a este:



Observe la estructura de la tabla intermedia. ¿Cuál es su clave primaria? ¿Hacia dónde van las claves ajenas? ¿Porqué los valores de los índices asociados a las claves ajenas nos están marcados como únicos?

- c. Modifique el fichero data.sql para insertar datos en la nueva tabla intermedia de manera que la diabetes pueda darse en perros y gatos, y el COVID solamente en perros. Compruebe que los datos se añadidos a la BD usando la consola de H2.
- d. Modifique el editor de enfermedades, de manera que permita seleccionar el conjunto de Tipos de mascotas para las que puede aparecer la enfermedad. Use una etiqueta con el texto “Pet types with known prevalence”. Para la generación de los checkboxes puede usar la etiqueta `<form:checkboxes>`. El resultado debería ser similar al que se muestra en la figura:

[HOME](#)
[FIND OWNERS](#)
[VETERINARIANS](#)
[VADEMECUM](#)
[LOGIN](#)
[REGISTER](#)

### Disease

Name

Description

La diabetes en perros es una enfermedad compleja causada por la falta de insulina o la respuesta inadecuada de esta. Cuando la mascota come, su sistema digestivo rompe los alimentos en varios componentes, incluyendo la glucosa, que es transportada a las células por la insulina, una hormona que segrega el páncreas. Cuando el animal no produce insulina o no puede utilizarla con normalidad, sus niveles de azúcar en sangre se elevan. El resultado es la hiperglucemia que si no se trata puede causar complicaciones.

Pet types with known prevalence:
☐ bird
☒ cat
☒ dog
☐ hamster
☐ lizard
☐ snake

Update Disease

[Pivotal](#)

- (Puede usar la [propiedad delimiter](#) para conseguir una separación entre los elementos generados)
- e. Modifique la vista de listado de enfermedades para añadir una nueva columna que muestre los tipos de mascotas donde la enfermedad se presenta como una lista no ordenada. El resultado debería ser similar al que se muestra en la siguiente figura:

[HOME](#)
[FIND OWNERS](#)
[VETERINARIANS](#)
[VADEMECUM](#)
[LOGIN](#)
[REGISTER](#)

Disease updated successfully!

### Diseases

Name	Description	Pet Types
COVID-19	Es una enfermedad infecciosacausada por un coronavirus recientemente. De acuerdo a los Centros para el Control y la Prevención de Enfermedades de los Estados Unidos, algunas mascotas — incluyendo perros y gatos — también se han infectado con el virus que causa la COVID-19. Sin embargo, en base a la información limitada que existe, se considera poco el riesgo de que los animales transmitan la COVID-19 a la gente."	<ul style="list-style-type: none"> <li>dog</li> </ul>
Diabetes	La diabetes en perros es una enfermedad compleja causada por la falta de insulina o la respuesta inadecuada de esta. Cuando la mascota come, su sistema digestivo rompe los alimentos en varios componentes, incluyendo la glucosa, que es transportada a las células por la insulina, una hormona que segrega el páncreas. Cuando el animal no produce insulina o no puede utilizarla con normalidad, sus niveles de azúcar en sangre se elevan. El resultado es la hiperglucemia que si no se trata puede causar complicaciones.	<ul style="list-style-type: none"> <li>cat</li> <li>dog</li> </ul>
Cambio de piel estacional	Algunos reptiles cambia de piel con el cambio de estaciones.	<ul style="list-style-type: none"> <li>snake</li> <li>lizard</li> </ul>

+Add disease

[Pivotal](#)

**Ejercicio 8. Nuestra primera regla de negocio compleja: Un diagnóstico solo puede ser de una enfermedad compatible con el tipo de mascota para el que se realiza.**

*Al tratarse de un ejercicio con varias alternativas de solución proporcionaremos un commit distinto por cada una de esas alternativas.*

**Historia de Usuario:** “ Como veterinario, **deseo que el sistema** me avise cuando intente emitir un diagnóstico de una enfermedad que no puede darse en el tipo de mascota para la que se emite el diagnostico, **para** evitar diagnósticos erróneos”

**Escenario Negativo:** Un veterinario crea un diagnóstico de otitis para una mascota que es un pez (que no tiene orejas ni oídos).

Este ejercicio puede resolverse de varias maneras distintas:

- a. La opción es crear un validador específico para diagnósticos que fuerce el cumplimiento de la regla de negocio configurado en el controlador directamente.

**Commit Solución:** [bcd7de938d6166d50ffbe0b2d3c9322d596a421](#)

- i. Cree una clase llamada DiagnoseDiseaseValidator con una estructura similar a PetValidator, que comprueba primero si la visita asociada al diagnóstico es nula, y si no lo es, que comprueba que la enfermedad diagnosticada está en el conjunto de enfermedades posibles del tipo de mascota de la visita.
- ii. Modifique el controlador de diagnósticos para que se compruebe si los diagnósticos son válidos y si no lo son vuelva a mostrar el formulario (no olvide pasar la información necesaria para que la vista pueda mostrar el formulario, como por ejemplo el conjunto de enfermedades, y veterinarios).
- iii. Configure el validador en el controlador de diagnósticos para que valide el diagnóstico usando la anotación @InitBinder. Fíjese en cómo se hace en el PetController y configúrelo de una manera similar.
- iv. Modifique el formulario de edición de diagnóstico para que muestre los errores asociados a la enfermedad.
- v. Ejecute la aplicación en modo depuración, y ponga un breakpoint en el método validate del validador. Compruebe si se está invocando al validador y si éste está forzando el cumplimiento de la restricción (puede probarlo creando un diagnóstico de COVID para Leo, el gato de George Franklin, al tratarse de un gato, con la configuración establecida en el ejercicio anterior, ese diagnóstico es imposible). ¿Por qué no genera errores el validador? ¿Cuál es la causa raíz de que no los genere?, ¿qué estamos haciendo mal en el controlador?
- vi. ¿Qué pasa si añadimos la anotación @Valid al parámetro del método save de la clase DiagnoseService? ¿Por qué no se está invocando en este caso a la validador personalizado?
- vii. ¿Se le ocurre alguna manera de modificar el formulario de edición de diagnósticos o el controlador de diagnósticos para que el validador funcione adecuadamente? (Observe los cambios que forman parte del commit de la solución para esta alternativa de implementación de la regla de negocio).

- viii. ¿Qué ocurre cuando hacemos un diagnóstico con una enfermedad válida, pero sin descripción? ¿Qué debería ocurrir? ¿Por qué no ocurre lo que debería ocurrir? ¿Se le ocurre alguna manera de solucionarlo? (Intente que dicha solución sea la más elegante y extensible posible).<sup>1</sup>
- b. Validar el cumplimiento de la regla de Negocio como parte de la operación save del servicio de diagnósticos, y lanzar una excepción específica que será capturada por el controlador.

**Commit Solución:** [c71e2153b3d40a5df0f7e94f1144a503dc5361ef](#)

- i. Comente el código que registra el InitBinding en el controlador de diagnósticos, para que no se aplique la solución a).
- ii. Cree una clase llamada ImpossibleDiseasesException que extienda a la clase Exception. La clase debe ubicarse en un nuevo paquete: org/springframework/samples/petclinic/model/businessrulesexceptions. Su constructor debe tomar dos parámetros un nombre de enfermedad y un tipo de mascota (los dos serán de tipo String). Evidentemente la clase debe tener propiedades para almacenar esos valores. En el constructor invoque al constructor de la clase padre con un mensaje similar a: "According to our vademecum a "+petType+" cannot develop "+disease.
- iii. Modifique el método save del servicio para que realice la validación de la regla de negocio (separe el código de la validación en un método auxiliar dentro de la misma clase). No olvide cambiar la signatura del método para reflejar que este método puede lanzar una excepción de tipo ImpossibleDiseaseException.
- iv. Capture la excepción desde el método de creación de diagnósticos del controlador de diagnósticos de manera que añada el error al BindingResult y se redirija al formulario de edición (no olvide pasar a la vista todos los datos necesarios para poder mostrarla, incluyendo el listado de enfermedades y veterinarios). Puede usar la función rejectValue del BindingResult para añadir el error al campo apropiado (en este caso "disease").
- v. Compruebe que se está aplicando la validación al intentar crear un diagnóstico imposible (puede probarlo creando un diagnóstico de COVID para Leo, el gato de George Franklin, al tratarse de un gato, con la configuración establecida en el ejercicio anterior, ese diagnóstico es imposible). Pruebe a crear un diagnóstico sin descripción ¿qué ocurre?
- vi. ¿Sería posible que en algún caso se llegara a grabar la información sin pasar por el método save y por tanto que algún programador de nuestro equipo se saltase la restricción inadvertidamente?<sup>2</sup>
- c. Creación de una anotación de validación y un validador. Esta solución consiste en crear dos ficheros java: una clase validadora, muy similar al que se creó para la opción a), y un interfaz que define una nueva anotación que podremos aplicar en nuestro código. Una vez tenemos creados ambos ficheros correctamente, podremos aplicar esa anotación a nuestras clases para que el propio sistema de

---

<sup>1</sup> Una pequeña pista, mire cómo se implementa la invocación a los validadores en <https://www.baeldung.com/javax-validation>

<sup>2</sup> A este respecto es interesante la siguiente lectura: <https://codete.com/blog/5-common-spring-transactional-pitfalls/> y especialmente el pitfall 1.



validación de Spring aplique las validaciones por nosotros, al igual que hace con las anotaciones `@Size` o `@NotNull`. Concretamente, crearemos una anotación que compruebe que la enfermedad del diagnóstico esté contenida en el conjunto de enfermedades posibles del tipo de mascota al que va asociado.

**Commit Solución:** [384e23fe5dff5bc8c2d5b73bdd5f698c317cec14](#)

- i. Comente la llamada a la función auxiliar de validación en el servicio de diagnósticos.
- i. Cree un interfaz java para representar a la anotación que vamos a crear. Dicho interfaz debe llamarse `ValidatePossibleDisease` y de ubicarse en un nuevo paquete a crear llamado:  
`org/springframework/samples/petclinic/service/businessrules`. Anote la interfaz con las siguientes anotaciones:  
`@Documented`  
`@Target(ElementType.TYPE)`  
`@Retention(RetentionPolicy.RUNTIME)`  
`@Constraint(validatedBy = {PossibleDiseaseValidator.class})`
- ii. Coloque una `@` justo antes de la `i` de interface para indicar que este interfaz especifica una nueva anotación. Por tanto, la signatura de la declaración debería ser: `@interface ValidatePossibleDisease`
- iii. El cuerpo de la interfaz debería ser:

```
String message() default "According to our vademecum such pet type  
cannot develop that disease";
```

```
Class<?>[] groups() default {};
```

```
Class<? extends Payload>[] payload() default {};
```

- iv. Cree una clase Java llamada `ElementInCollectionValidator` en el paquete:  
`org/springframework/samples/petclinic/service/businessrules`. Esta clase debe implementar `ConstraintValidator<ConfirmPassword, Object>`
  - v. Anote la clase `Diagnose` con la anotación `@ValidatePossibleDisease`
  - vi. Compruebe que se está aplicando la validación al intentar crear un diagnóstico imposible (puede probarlo creando un diagnóstico de COVID para Leo, el gato de George Franklin, al tratarse de un gato, con la configuración establecida en el ejercicio anterior, ese diagnóstico es imposible). Pruebe a crear un diagnóstico sin descripción ¿qué ocurre?
  - vii. ¿Se muestra el mensaje de validación asociado al validador en el formulario? ¿Por qué?
  - viii. Inserte código que permita mostrar los mensajes globales de validación (lo no asociados a algún campo concreto) en el formulario de creación de diagnósticos. Puede usar la anotación `.`
- d. ¿Cuál cree usted que es la mejor opción desde el punto de vista del diseño de software (cohesión, acoplamiento, simplicidad, resusabilidad, legibilidad del código, etc.)? ¿Por qué? ¿Cree que esa opción es generalizable para otras reglas de negocio similares a validar? ¿Cómo la generalizaría?
  - e. ¿Se le ocurre alguna otra manera de asegurar que no se creen diagnósticos imposibles mediante el formulario?